Examiner's Amendment and Statement of Reasons for Allowance

1. This action is responsive to Applicant's amendment filed July 28, 2008.

Examiner's Amendment

2. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Mr. Steven J. Hanke, Registration Number 58,076, on October 09, 2008 for obviating any potential 101 issues and put the claims in condition for allowance.

The application has been amended as follows:

1. (Currently Amended) A system for automatically generating a hierarchical register consolidation structure, comprising:

a processor;

a graph generator that parses a Hardware Description High level Design Language (HDL) file in three different passes to generate an intermediate graph containing definitions of microprocessor-accessible registers, node interrelationships and summary bits, bit offsets and masks associated with alarm registers of external devices identified by said HDL file, wherein:

a first pass extracts information associated with said microprocessor-accessible registers;

a second pass identifies said node interrelationships and associates said alarm registers with mask registers and persistency and delta information; and

a third pass generates said bit offsets and said masks associated with said alarm registers and associates said summary bits therewith;

a graph converter, associated with said graph generator, that selectively adds virtual elements and nodes to said intermediate graph to transform said intermediate graph into a mathematical tree; and

a description generator, associated with said graph converter, that employs said mathematical tree to generate a static tree description ~~to form a hierarchical register consolidation structure to provide a logical representation of said microprocessor-accessible registers, node interrelationships, summary bits and masks of said external devices~~ in a programming language suitable for use by a device-independent condition management structure and an HTML traversable tree representation based on said mathematical tree, wherein both of said static tree description and said HTML traversable tree representation form a hierarchical register consolidation structure to provide a logical representation of said microprocessor-accessible registers, node interrelationships, summary bits, bit offsets, and masks of said external devices.

2. (Currently Amended) The system as recited in Claim 1 wherein said programming language is C ~~intermediate graph further contains bit offsets associated with said alarm registers~~.

3. (Currently Amended) The system as recited in Claim 1 wherein said HDL file is produced by a hardware description tool ~~description generator further generates an HTML traversable tree representation based on said mathematical tree~~.

4. (Currently Amended) The system as recited in Claim 1 [[7]] wherein said first pass extracts names, addresses, definitions, bit positions used and bit position names of said microprocessor-accessible registers.

6. (Currently Amended) The system as recited in Claim 1 [[3]] wherein said condition management structure manages interrupts associated with said external devices employing said static tree or said HTML traversable tree.

7. (Currently Amended) The system as recited in Claim 1 [[2]] wherein said hierarchical register consolidation structure pertains to a real-time system ~~graph generator parses said HDL file in three different passes to generate said intermediate graph wherein a first pass extracts information associated with said microprocessor-accessible registers; a second pass identifies said node interrelationships and associates said alarm registers with mask registers persistency and delta information; and a third pass generates said bit offsets and said masks associated with said alarm registers and associates said summary bits therewith.~~

8. (Currently Amended) A method of automatically generating a hierarchical register consolidation structure, comprising:

parsing a Hardware Description High level Design Language (HDL) file in three passes to generate an intermediate graph containing definitions of microprocessor-accessible registers, node interrelationships and summary bits, bit offsets and masks associated with alarm registers of external devices identified by said HDL file, wherein:

a first pass extracts information associated with said microprocessor-accessible registers;

a second pass identifies said node interrelationships and associates said alarm registers with mask registers and persistency and delta information; and

a third pass generates said bit offsets and said masks associated with said alarm registers and associates said summary bits therewith;

selectively adding virtual elements and nodes to said intermediate graph to transform said intermediate graph into a mathematical tree; and

employing said mathematical tree to generate a static tree description in a programming language suitable for use by a device-independent condition management structure and an HTML traversable tree representation based on said mathematical tree, wherein both of said static tree description and said HTML traversable tree representation form a hierarchical register consolidation structure to provide a logical representation of said microprocessor-accessible registers, node interrelationships, summary bits, bit offsets, and masks of said external devices.

9. (Currently Amended) The method as recited in Claim 8 wherein said programming language is C ~~intermediate graph further contains bit offsets associated with said alarm registers~~.

10. (Currently Amended) The method as recited in Claim 8 wherein said first pass extracts names, addresses, definitions, bit positions used and bit position names of said microprocessor-accessible registers ~~further comprising employing said static tree description to generate an HTML traversable tree representation based on said mathematical tree~~.

Claims 15-20 Canceled.

Examiner's Statement of Reason(s) for Allowance

3. Claims 1-14 are allowed.

4. The following is an examiner's statement of reasons for allowance:

The prior arts of record: **Nelson** et al., teaches each device is represented by a node in a hierarchical tree, referred to herein as an interrupt source tree (IST). The root and intermediate nodes of the IST represent dispatching or determining points; this removes the need for the device drivers to contain dispatching code; the drivers now only need to contain interrupt handling code specific to the device itself. At these nodes code is executed to determine which branch to take down the IST. The bottom nodes are the leaf nodes; the leaf nodes identify the device interrupt handler. **Gregory** et al., teaches a digital circuit is synthesized from a text description of a digital system. During synthesis, a parse tree with parse nodes is constructed and retained. The relationship between the parse nodes and the circuit elements synthesized from those parse nodes is retained. Using that relationship, analysis results associated with circuit elements can be related to the text that generated those circuit elements. **Aleksic** et al., teaches an integrated circuit modeling system facilitates automatic design of register based hardware devices by generating major pieces of the development outputs from a single input, such as a single register specification source file. The modeling code is kept coherent for all major phases of design and testing. The register specification source file contains all the register information about the device being developed. **Kodosky** et al., teaches a computer-implemented system and method for generating a hardware implementation of graphical code. The method may operate to configure an instrument to perform measurement functions, wherein the instrument includes a programmable hardware element. The method comprises

first creating a graphical program, wherein the graphical program may implement a measurement function. The configured hardware element thus implements a hardware implementation of the second portion of the graphical program. During generation of the hardware implementation, the computer system may operate to estimate and/or display one or more of size and cost of a hardware implementation of the graphical program. **Williams** (732) et al., teaches method and system for instrumenting test case execution processing of a hardware description language (HDL) model using a simulation control program. In accordance with the method of the present invention, a set name application program interface (API) entry point is called wherein the set name API entry point includes program instructions for naming a simulation control program in association with test case execution of the HDL model. **Matsunaka** et al., teaches an apparatus and method for translating a function description of a circuit presented in hardware description language includes parsing the function description of the circuit to generate a parse tree. The structure of the parse tree is deformed to optimize the test level redundancy of the function description to thereby generate a deformed parse tree. **Johnson**, teaches an invention is directed to a method of processing a database comprising information regarding hardware design language ("HDL") events occurring during a simulation of a hardware design. The method comprises identifying in the database all HDL events comprising observability events; obtaining from each of the identified observability HDL events information pertaining to a signal driving the identified observability HDL event observed on an observability bus; and creating a data structure comprising a plurality of entries, wherein each of the entries corresponds to one of the signals observed on the observability bus and contains signal

information pertaining to the one of the observed signals. **Killian** et al., teaches an automated processor design tool uses a description of customized processor instruction set extensions in a standardized language to develop a configurable definition of a target instruction set, a Hardware Description Language description of circuitry necessary to implement the instruction set, and development tools such as a compiler, assembler, debugger and simulator which can be used to develop applications for the processor and to verify it. The structure of the parse tree is deformed to optimize the test level redundancy of the function description to thereby generate a deformed parse tree. **Miller** et al., teaches a method of designing an integrated circuit using a general purpose programming language can include identifying a number of instances of each class allocated in a programmatic design implemented using the general purpose programming language and modeling the global memory of the programmatic design. A data flow between the modeled global memory and instructions of the programmatic design which access object fields can be determined and access to the modeled global memory can be scheduled. The programmatic design can be translated into a hardware description of the integrated circuit using the modeled global memory, the data flow, and the scheduled memory access. **Ly** et al., teaches programmed computer generates descriptions of circuits (called "checkers") that flag functional defects in a description of a circuit undergoing functional verification. The programmed computer automatically converts the circuit's description into a graph, automatically examines the graph for instances of a predetermined arrangement of nodes and connections, and automatically generates instructions that flag a behavior of a device represented by the instance in conformance with a known defective behavior. The checkers can be used during simulation or

emulation of the circuit, or during operation of the circuit in a semiconductor die The circuit's description can be in Verilog or VHDL and the automatically generated checkers can also be described in Verilog or VHDL. Ho et al., teaches a programmed computer searches for functional defects in a description of a circuit undergoing functional verification in the following manner. The programmed computer simulates the functional behavior of the circuit in response to a test vector, automatically restores the state of the simulation without causing the simulation to pass through a reset state, and then simulates the functional behavior of the circuit in response to another test vector. A predetermined rule can be used to identify test vectors to be simulated, and the predetermined rule can depend upon a measure of functional verification, including the number of times during simulation when a first state transition is performed by a first controller at the same time as a second state transition is performed by a second controller. **Ho** et al., teaches programmed computer searches for functional defects in a description of a circuit undergoing functional verification in the following manner. The programmed computer simulates the functional behavior of the circuit in response to a test vector, automatically restores the state of the simulation without causing the simulation to pass through a reset state, and then simulates the functional behavior of the circuit in response to another test vector. A predetermined rule can be used to identify test vectors to be simulated, and the predetermined rule can depend upon a measure of functional verification, including the number of times during simulation when a first state transition is performed by a first controller at the same time as a second state transition is performed by a second controller. **Chen** et al., teaches a method and apparatus for detecting and decomposing component loops in a logic design is described. The

invention first detects any component loops when the compiler schedules the processing order of the combinational logic components in the digital circuit design. To identify component loops, the compiler levelizes the design and sorts the combinational logic components, making sure that no true combinational logic loops exist. **Williams** (508), teaches a method and apparatus for developing placement characteristics of a circuit design in conjunction with developing functional aspects of the circuit. In various embodiments, an application programming interface (API) is programmed in a hardware definition language (HDL). The API provides placement directives that can be called from the HDL code that defines functional characteristics of the circuit. The API can also be used in a test bench in order to analyze both the functional and physical placement characteristics of the design. Since the API is programmed in HDL, the placement generated during the implementation phase is the same as the placement analyzed during functional simulation. **Broughton** et al., teaches a method for compiling a cycle-based design involves generating a parsed cycle-based design from the cycle-based design, elaborating the parsed cycle-based design to an annotated syntax tree, translating the annotated syntax tree to an intermediate form, and converting the intermediate form to an executable form. **Meribout,** teaches a front-end compiler 103 carries out a syntax analysis of a description file describing a desired electronic circuit model with a predetermined high level description language, to generate a control data flow graph having a predetermined graph structure. A back-end compiler divides the control data flow graph into threads composed of a set of a plurality of connected nodes and achieving a particular function. The back-end compiler 105 optimizes the divided threads to meet with a predetermined area restriction and a predetermined waiting

time restriction, to obtain designation information of the number, the function, the placement and routing of logic cells for the desired electronic circuit model. And new prior arts put in record: US Patent No. 7,024,660 by **Andrade** et al., teaches a system and method for debugging a program which is intended to execute on a reconfigurable device. A computer system stores a program that specifies a function, and which is convertible into a hardware configuration program (HCP) and deployable onto a programmable hardware element comprised on the device. The HCP is generated based on the program, specifies a configuration for the programmable hardware element that implements the function, and further specifies usage of one or more fixed hardware resources by the programmable hardware element in performing the function. US Patent No. 7,290,244 by **Peck** et al., teaches system and method for configuring a reconfigurable I/O (RIO) device to perform a function in response to user requirements. A graphical user interface program receives user input specifying a function. A configuration generation program generates a hardware configuration program based on the user input. The hardware configuration program is usable to configure a device to perform the function, where the device includes a programmable hardware element and one or more fixed hardware resources coupled to the programmable hardware element. The hardware configuration program is deployable onto the programmable hardware element and specifies usage of the fixed hardware resources by the programmable hardware element in performing the function. And US Patent No. 7,085,670 by **Odom** et al., teaches system and method for configuring a device to perform a function, where the device includes a programmable hardware element and one or more fixed hardware resources. A program is stored which represents the function. A hardware configuration

program is generated based on the program, specifying a configuration for the programmable hardware element that implements the function, and usage of the fixed hardware resources by the programmable hardware element in performing the function. A deployment program deploys the hardware configuration program onto the programmable hardware element, where, after deployment, the device is operable to perform the function, where the programmable hardware element directly performs a first portion of the function, and the programmable hardware element invokes the fixed hardware resources to perform a second portion of the function. US Patent No. 5,541,850 by **Vander** et al., teaches set of circuit specifications including an internal memory structure is developed and then described in a hardware description language that is entered into a computer system. The circuit description is then synthesized on the computer to form a netlist to specify the circuit. From this netlist, an integrated circuit is produced on a semiconductor die, which is then packaged for use. A method for synthesizing a netlist from a hardware description including an internal memory structure includes converting the hardware description into an internal signal list, which contains an indication of the presence of an internal memory structure in the described circuit. US Patent No. 7,000,213 by **Banerjee** et al., teaches Digital circuit is synthesized from algorithm described in the MATLAB programming language. A MATLAB program is compiled into RTL-VHDL, which is synthesizable using system-specific tools to develop ASIC or FPGA configuration. Intermediate transformations and optimizations are performed to obtain highly optimized description in RTL-VHDL or RTL Verilog of given MATLAB program. Optimizations include levelization, scalarization, pipelining, type-shape analysis, memory optimizations, precision analysis and scheduling. US

Patent No. 5,537,580 by **Giomi** et al., teaches method for fabricating an integrated circuit includes the steps of: (a) describing the functionality of an integrated circuit in terms of a behavioral hardware description language, where the hardware description language describes behavior which can be extracted as a state machine; (b) extracting a register level state machine transition table of the state machine from the hardware description language; (c) generating a logic level state transition table representing the state machine from the register level state machine description; (d) creating a state machine structural netlist representing the state machine from the logic level state transition table; and (e) combining the state machine structural netlist with an independently synthesized structural netlist to create an integrated circuit structural netlist including the state machine to provide a basis for chip compilation, mask layout and integrated circuit fabrication. The method results in a synchronous state machine being extracted from an register-transfer (RT) level representation taken from a scheduled behavioral hardware description language description such as a Verilog or VHDL. US Patent No. 7,389,496 by **Eckhart** et al., teaches For use with a processor employing a hierarchical register consolidation structure (HRCS), a condition management system and method of operation thereof. In one embodiment, the system includes a condition management structure (CMS) that abstracts groups of status indicators associated with the HRCS into a tree of hierarchical container objects and element objects. Each of the container objects is associated with at least one of the element objects and linked to a single parent object, and each of the element objects configured to represent at least one of the status indicators and linked to a single child object. The system further includes an abstraction retrieval subsystem that employs the

CMS to traverse the HRCS to determine a condition of at least one of the status indicators, and an abstraction management subsystem that employs the CMS to control a propagation of selected ones of the status indicators through the HRCS. However, none of them, taken alone or in combination, teaches the features in such a manner as recited in independent claims 1, and 8.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

5. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 8:00am - 4:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Chih-Ching Chow/
Examiner, Art Unit 2191
10/13/08

/Wei Y Zhen/
Supervisory Patent Examiner, Art Unit 2191